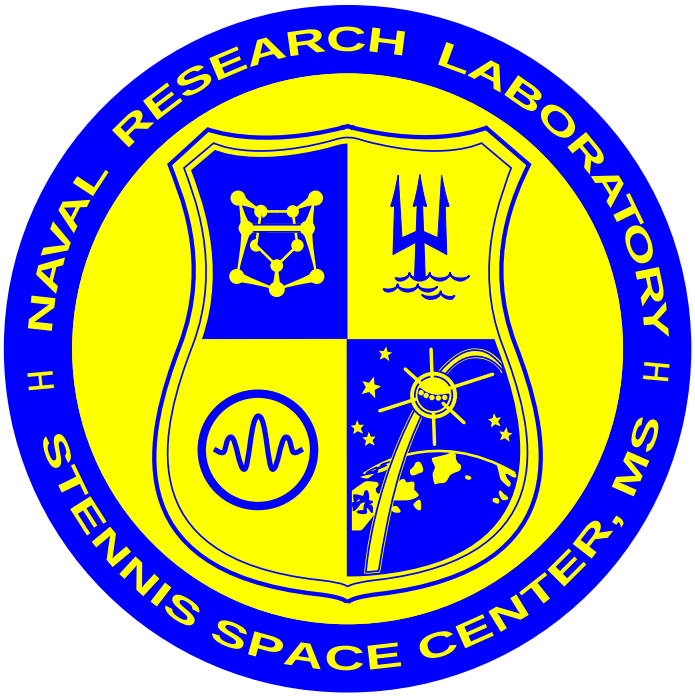
Neptune Sciences, Inc.
40201 Highway 190, Slidell, LA 70461

Automated Processing System User's Guide
Version 2.8



Paul Martinolich
6 September 2004

Automated Processing System
User's Guide
Version 2.8

Paul Martinolich

Neptune Sciences, Inc.

ABSTRACT

The Automated Processing System (APS) is a collection of UNIX programs and shell scripts designed to generate map-projected image data bases of satellite derived products from a large flow of raw satellite input data in an automated fashion. Individual scenes are sequentially processed from the raw digital counts (Level−1) using standard parameters to a radiometrically and geometrically corrected (Level−3) product within several minutes. It further processes the data into four different temporal (daily, 8-day, monthly, and yearly) composites or averages (Level−4). These products are stored in the Hierarchical Data Format (HDF). Additionally, it automatically generates quick-look ''browse'' images in JPEG format and may populate an SQL database using PostgreSQL.

The APS was designed for the Naval Research Laboratory Remote Sensing Applications Branch at Stennis Space Center, MS (now the Ocean Sciences Branch) to handle the continuous stream of satellite data. Originally, the system was designed to produce sea surface temperature maps from data collected by the Advanced Very High Resolution Radiometer (AVHRR). Data from the AVHRR is received daily (up to six passes per day). The APS provides for near real-time processing with the option of reprocessing historical data. The APS has since been upgraded to process ocean color satellite data from SeaWiFS and MODIS (Terra).

The APS uses a simple monitoring technique, which has been found to be fairly reliable. The main driver regularly polls a specified input directory for incoming data and, for each found, executes what are known as *areas* scripts on the file in a working directory. After each area script has been run on the file, it is moved to an output directory. This method uses the directory as the queueing system for data to be processed. The areas scripts do the actual construction of the desired results (i.e. the data bases).

The 2.8 version represents the processing algorithms employed at the Naval Research Laboratory as of 6 September 2004 for the SeaWiFS, MOS, MODIS, and AVHRR sensors. The system has been tested on RedHat Linux 7.1 and SGI IRIX 6.5.

7 September 2004

Automated Processing System
User's Guide
Version 2.8

Paul Martinolich

Neptune Sciences, Inc.

1. INTRODUCTION

The Automated Processing System (APS) is a collection of UNIX programs and shell scripts designed to generate map-projected image data bases of satellite derived products from a large flow of raw satellite input data in an automated fashion. Individual scenes are sequentially processed from the raw digital counts (Level–1) using standard parameters to a radiometrically and geometrically corrected (Level–3) product within several minutes. It further processes the data into four different temporal (daily, 8-day, monthly, and yearly) composites or averages (Level–4). These products are stored in the Hierarchical Data Format (HDF). Additionally, it automatically generates quick-look ''browse'' images in JPEG format and may populate an SQL database using PostgreSQL.

The APS was designed for the Naval Research Laboratory Remote Sensing Applications Branch at Stennis Space Center, MS (now the Ocean Sciences Branch) to handle the continuous stream of satellite data. Originally, the system was designed to produce sea surface temperature maps from data collected by the Advanced Very High Resolution Radiometer (AVHRR). Data from the AVHRR is received daily (up to six passes per day). The APS provides for near real-time processing with the option of reprocessing historical data. The APS has since been upgraded to process ocean color satellite data from SeaWiFS and MODIS (Terra).

The APS uses a simple monitoring technique, which has been found to be fairly reliable. The main driver regularly polls a specified input directory for incoming data and, for each found, executes what are known as *areas* scripts on the file in a working directory. After each area script has been run on the file, it is moved to an output directory. This method uses the directory as the queueing system for data to be processed. The areas scripts do the actual construction of the desired results (i.e. the data bases).

The 2.8 version represents the processing algorithms employed at the Naval Research Laboratory as of 6 September 2004 for the SeaWiFS, MOS, MODIS, and AVHRR sensors. The system has been tested on RedHat Linux 7.3 and SGI IRIX 6.5.

1.1. What You Need to Know to Get Started

Before starting to use the APS, you should be familiar with your UNIX environment, Bourne shell programming, and remote sensing, especially with regard to computer processing of satellite data.

1.2. What is the Automated Processing System?

The APS is a collection of programs designed to allow scientists to generate co-registered image data bases of geophysical parameters derived from remotely sensed data. The system is characterized by: *extension* and *automation*.

Extension is the use of small programs, each designed to address a specific task, and a shell to ''glue'' them together. This idea is similar to the UNIX operating system and its many programs like `cat`, `tr`, `basename`, etc. and allows the user to augment the system with their own features and programs.

Automation is the technique of making a system operate without human effort or decision. For the APS, it is achieved by setting up a directory structure and using a script to monitor the directories for new input data. As new data is made available to the system, it is processed – all without the user's intervention.

To address these characterizations, all programs within the APS must not ask the user to make decisions during execution. All user input *must* be provided to the program upon *start*. Thus, the APS does not contain any GUIs or visualization programs.

1.3. Understanding the APS

The APS is not a shell language or generic batch processor. It is a collection of programs used to process satellite data. Specifically, the system contains programs for

- atmospheric correction
- geometric registration
- image processing
- compositing

Each of these programs provides the scientist with the tools to manipulate and study satellite data.

Additionally, the system includes programs that allow the above scientific programs to be used in an unattended way. These include functions to:

- query data files for information
- monitor the system for new data

2. GETTING STARTED

To get started using the APS, we will write an areas script in this section. This script will be installed into the APS in the next section called ‘‘AUTOMATION’’ The reason for some code constructs will become clear in that chapter.

In this section, we will create a Bourne shell script for the creation of chlorophyll *a* image maps for waters within the Gulf of Mexico. It is assumed that the APS has been installed at your site. If the system has not been installed, see section 5 titled, ‘‘INSTALLING APS’’ near the end of this document. Furthurmore, it is assumed that APS was installed in `/home/aps/aps_v2.8` and that the variable `AUTO_DIR` is set to `/home/aps/aps_v2.8`. If you can't wait to create an area of interest, see section 9, "PREDEFINED MAPS", which are already available.

2.1. Defining Your Interest

Before writing a script, we must decide what it is we want to create. In this example, the objective is to create an image data base of chlorophyll *a* concentration using two different algorithms for waters within the Gulf of Mexico. We wish to compare the OC2 algorithm with the OC4 algorithm in both open ocean and coastal waters. We plan to use the data from the SeaWiFS sensor.

2.2. Waters Within the Gulf of Mexico

First, we will our region of interest. For this example, it will be defined as the waters of the Gulf of Mexico. Using a USGS or National Geographic map of the Gulf of Mexico, we decide to define this region as the area within the following geographical boundaries:

- The upper left corner shall be 31 N and 98 W
- The lower right corner shall be 17 N and 80 W

Because we plan to do spatial and temporal variability studies of the loop current in the Gulf of Mexico (and maybe make a nice movie loop of several days worth of images), we need each scene to be registered to a standard map projection. We use the term *image map* to indicate a specific map projected image. An *image map* is the combination of a map projection and an output image. Each pixel's relationship to its neighbors is defined by the projection used. In this example, we want our image to be a full-resolution image of chlorophyll *a* represented as a Mercator projection. The Mercator projection as one of thirty available map projections provided by the General Cartographic Transformation Package. The GCTP package was obtained from the United States Geological Survey.

One group of outputs generated by the Level−2 APS programs is collectively termed a control points structure. This structure allows a program to determine the geographical coordinates of any pixel in the original scene by tying down specific points in the satellite scene (sample,line) to geographical coordinates (longitude,latitude). These points are commonly called Ground Control Points (GCP). In the APS, the structure forms a regular grid across the image. That is, the for each column and row in the input image, a known latitude and longitude pair is given. Thus, the column or pixel locations as well as the row or line locations are given as 1-dimensional arrays. The pixel and line locations are stored in separate arrays. The latitudes and longitudes at the intersections of the row and columns are stored in 2-dimenional arrays. The latitudes and longitudes are also stored in separate arrays. This information is used by the geometric registration program.

The APS includes a geometric registration program known as imgMap. This program moves the pixels or data from the input file to newer locations in the output file. This process is known as geometric registration or ‘‘warping.’’ The translation from input pixel to output pixel is defined by the *image map* and the control points structure. The results of the pixel movement will depend up the accuracy of the input control points structure. Additionally, the parameters unique to the selected map projection will effect the accuracy in scale of the resulting image. For example, the Mercator map projection will exaggerate land mass as we reach the poles. That is, Greenland will appear much larger in size than it actually is.

2.3. Controlling the Projection Characteristics of the Output

The *image map* parameters determine the projection of the output image as well as of the imaginary map of the world of which it is a part. Although the image is projected onto that map, the input parameters allow the user to control where the window will be positioned over the map as well as the scale of the map.

The first two required parameters will associate a longitude and latitude (LL_1) from the world to a particular location on the output image (XY_1). The XY_1 parameter refers to the image coordinates (sample and line location) of a point in the image whereas LL_1 refers to the geocoordinates (longitude and latitude) of a point on the world map. The points will be associated so that the image point (XY_1) overlays the map point (LL_1). It may be convenient to choose LL_1 to be the center of the image by assigning (XY_1) to (number of samples/2,number of lines/2). Alternately, the user may wish that a certain landmark appear at a certain location in the image. In such a case, the landmark's geocoordinates would be entered for LL_1 and the desired image location specified by XY_1.

The next two parameters will determine the scale of the *image map*. The first parameter is a second point on the world map (LL_2) and the second parameter is a distance (in pixels) away from the first (DELTA). The use of LL_2 and DELTA in conjunction with LL_1 and XY_1 controls the scale of the map and, hence, also controls how much of the mapped image appears within the image. LL_2 represents the geocoordinates of another point on the world map and DELTA represents the separation in pixels between that point and the XY_1 image location. A positive DELTA represents a horizontal separation, whereas a negative value represents a vertical separation.

Note that this second point need not be within the image and that the absolute value of DELTA may be larger than the image width or height. For given parameters, a larger absolute DELTA will decrease the geographical area covered by the window (enlarge the map or *increase* the scale); a smaller absolute value will increase this area (contract the map or *decrease* the scale). The direction of the second point relative to the first – that is, where they both fall on the world map – is determined solely by the projection.

Although DELTA represents the separation in either the horizontal and vertical direction (not the absolute separation), the points for LL_1 and LL_2 must be chosen such that they have both a horizontal *and* vertical separation on the imaginary map. Therefore, some *a priori* knowledge of where these points will fall on that map is required when choosing these parameter values.

A convenient way to determine the value for LL_2 and DELTA is to use the geocoordinates of another landmark for LL_2 and enter the desired separation between LL_1 and LL_2 for DELTA. Another convenient way to determine these values is to determine the scale for the map at LL_1. (In certain cases, depending on the projection, the scale will vary greatly even within the image area.) That is, the user decides how many image pixels (DELTA) should separate a longitudinal or latitudinal degree and assign LL_2 accordingly. For example, if the scale at LL_1 is to be one latitudinal degree per 100 pixels and LL_1 is 10 degrees latitude and 38 degrees longitude, LL_2 would be 9 and 38 degrees and DELTA would be -100 (assuming that north is on the top of this projection).

2.4. Creating the Map

In this example, we align the North West corner (LL_1) with the upper left corner of the image (XY_1). Thus, LL_1 will have a value of 98 W and 31 N and XY_1 will have a value of sample 1, line 1. (For images produced by **imgMap** (1) the origin of the image is located in the upper left. In other words, we count our lines down from the top and our samples (or columns) from the left. So, line 1 is at the top of the image; line *n* is at the bottom. Pixel 1 is on the left edge of the image; pixel *n* is on the right edge.)

To define the second point (which does not have to exist in the output image), we need to set the DELTA component. This component is the horizontal or vertical separation in pixels on the projected image between the LL_1 (the NW corner) and LL_2. This parameter, along with the latitude and longitude of the second point, define the resolution of the output image. To maintain a 0.01 degrees/pixel resolution in the longitudinal direction, we set DELTA to 1799. If pixel 1 corresponds to 98 W and pixel 1800 corresponds to 80 W, then there are 1800 - 1, or 1799, pixels between them.

For the time being we will assume that the scale in the latitudinal direction is the same as in the longitudinal direction, though for the Mercator projection this is certainly *not* true. Since we want to place LL_2 near 80 W and 17 N, we will start by making a temporary map with 1400 lines because the Northern

and Southern latitude differ by 14 degrees and we have a scale of 0.01 degrees/pixel.

Once the map has been defined we will use the **toll** command to get a better approximation. Remember that LL_1, XY_1, LL_2, and DELTA define the image to the imaginary map with LL_1 and XY_1 being tied together. The size of the window (our output image) over the imaginary map can be different in size.

The program **maps** (1) is used to create the *image map*. We give the mapped image a name and store the parameters in an HDF file for later use. We call this file a *map file*. Note that in the example below, we set the central meridian and latitude of true scale to the middle of our projection and that we use the WGS 84 datum (12). See **imgMap** (1) for a complete list of possible map projections and datums.

So we begin by running the **maps** program and attempt to create our first *image map*:

```
$ cd
$ pwd
/home/aps/
$ maps
maps>create temp
Map Projection (? for list) ? [ Mercator ]
Size of Map (w h) ? 1800 1400
Pt1 of Map (lon lat) ? -98.0 31.0
Pt1 of Map (x y) ? [ 1 1 ]
Pt2 of Map (lon lat)  ? -80.0 17.0
Horizontal/Vertical Separation in pixels between pt1 and pt2 ? [ 1799 ]
Aspect ratio ? [ 1.0 ]
Datum (? for list) ? [ WGS 84 ]
SMajor ? [ 6378137.000000 ]
SMinor/Eccentricity ? [ 6356752.314245 ]
Longitude of Central Meridian (SDDDMMMSSS.SSS) ? -089000000.000
Latitude of True Scale (SDDDMMMSSS.SSS) ? 024000000.000
False Easting ? [ 0.0 ]
False Northing ? [ 0.0 ]
maps>setmap temp
maps>toll 1 1
-98.000000 31.000000
maps>toll 1800 1400
-80.000000 18.241954
maps>toxy -80.0 17.0
1800.000000 1529.447826
```

From this last information, we will increase the size of the image (that is, the "window" over the imaginary map) to 1530. So to continue:

```
maps>create Gulf
Map Projection ? [ Mercator ]
Size of Map (w h) ? 1800 1530
Pt1 of Map (lon lat) ? -98.0 31.0
Pt1 of Map (x y) ? [ 1 1 ]
Pt2 of Map (lon lat) ? -80.0 17.0
Horizontal/Vertical Separation in pixels between pt1 and pt2 ? [ 1799 ]
Aspect ratio ? [ 1.0 ]
Datum (? for list) ? [ WGS 84 ]
```

```
SMajor ? [ 6378137.000000 ]
SMinor/Eccentricity ? [ 6356752.314245 ]
Longitude of Central Meridian (SDDDMMMSSS.SSS) ? -89000000.000
Latitude of True Scale (SDDDMMMSSS.SSS) ? 24000000.000
False Easting ? [ 0.0 ]
False Northing ? [ 0.0 ]
maps>setmap
Name:Gulf
maps>toll 1800 1530
-80.000000 16.994684
maps>delete temp
maps>list
Gulf
maps>save gulf_map.hdf
maps>quit
$ ls gulf_map.hdf
gulf_map.hdf
$ hdf gulf_map.hdf list
File: gulf_map.hdf

File Attributes: (5)

createTime = "Tue Sep  3 20:45:42 2002"
createAgency = "Naval Research Laboratory, Stennis Space Center"
createSoftware = "APS v2.8"
createUser = "martinol"
createPlatform = "i686-pc-linux-gnu"

Data Sets: (1)

float64 Gulf [29]
    createTime = "Tue Sep  3 20:45:42 2002"
    createUser = "martinol"
    createPlatform = "i686-pc-linux-gnu"
    productName = "Map Projection Parameters"
```

To process this region the newly created map file `gulf_map.hdf` should be copied to the APS `data` directory. By default, the APS has a default map file to hold all known maps named `maps.hdf`. It resides in the `data` directory. To use the default location, you can copy the `Gulf` map from the `gulf_map.hdf` file to the `maps.hdf` file. If not, then your scripts you will have to set the `MapFile` variable so that APS will know which file to look at to find the `Gulf` projection information.

To copy the `gulf_map.hdf` file to the data directory use the command (remember we will have to add `MapFile` to our script):

```
$ cp gulf_map.hdf ~/aps_v2.8/data
```

To copy the `Gulf` area from the `gulf_map.hdf` file to the standard APS maps file (`maps.hdf`), we will use the APS provided program `hdf`. Note that the name we select should not already be in the `maps.hdf file`. See **hdf** (1) for more information and examples.

```
$ hdf gulf_map.hdf copy ~/aps_v2.8/data/map.hdf Gulf
```

Now that the projection information is done, we can write an areas script that will process data. The script must be a UNIX text file, so use your favorite text editor (vi, emacs, nedit, jot, etc.). If you use an editor that automatically creates backups, then sure to delete that file from the `areas` directory. In this particular case, we are going to create our script in our home directory and use the `gulf_map.hdf` directly from there also. The script, which we save is `SwfGulf`, looks like:

```
#!/bin/ksh
 . $AUTO_DIR/bin/apsScripts.sh
 . $AUTO_DIR/bin/swfScripts.sh
MapName=Gulf
MapFile=$HOME/gulf_map.hdf
MapExt=GOM
swfProcess $1 $0
```

Note the "`$1`" in the last argument. It is required. The second argument "`$0`" is optional, but is very useful so add it.

Additionally, note that we have added the `MapFile` line, because we decided, in this example, to copy our file (`gulf_map.hdf`) to the `data` directory.

2.5. Selecting Our Products

Looking up the man page for **MSl12** (1) we see that this program can produce the chlorophyll products we are seeking. The documenation on MSl12 indicates that the products are named: "chl_oc2" and "chl_oc4".

If this study is specific only to the Gulf of Mexico, then we only need to list them in the `SwfGulf` script. To do that, we define a Bourne shell variable called `L3ProdList`. In this case, our script now looks like this:

```
#!/bin/ksh
 . $AUTO_DIR/bin/apsScripts.sh
 . $AUTO_DIR/bin/swfScripts.sh
MapName=Gulf
MapFile=$HOME/gulf_map.hdf
MapExt=GOM
L3ProdList="chl_oc2 chl_oc4"
swfProcess $1 $0
```

A second option is to specify the desired products by modifing the default product file (`$AUTO_DATA/seawifs/seawifs_def_l2prod.dat`). This file, however, is *global* to APS. That is, for all scripts that do not define `L3ProdList`, these products will be produced.

2.6. Selecting Browse Products

To create browse images, we must select our images by setting the `L3BrowseList` variable. Or, as above, we can modify the default browse file (`$AUTO_DATA/seawifs/seawifs_def_l2browse.dat`). Our script will now look like this:

```
#!/bin/ksh
 . $AUTO_DIR/bin/apsScripts.sh
 . $AUTO_DIR/bin/swfScripts.sh
```

```
MapName=Gulf
MapFile=$HOME/gulf_map.hdf
MapExt=GOM
L3ProdList="chl_oc2 chl_oc4"
L3BrowseList="chl_oc2 chl_oc4"
swfProcess $1 $0
```

By default, this version of APS will automatically generate temporal composites (daily, weekly, monthly, and yearly). The weekly composites are actually 8-day composites. As with the Level–3 products, the APS has default files for both the Level–4 products and their quick-look images. These are stored in the files `$AUTO_DATA/seawifs/seawifs_def_l4prod.dat` and `$AUTO_DATA/seawifs/seawifs_def_l2browse.dat`, respectively. To override these options, we add two additional lines to our script. Our script will now look like this:

```
#!/bin/ksh
 . $AUTO_DIR/bin/apsScripts.sh
 . $AUTO_DIR/bin/swfScripts.sh
MapName=Gulf
MapFile=$HOME/gulf_map.hdf
MapExt=GOM
L3ProdList="chl_oc2 chl_oc4"
L3BrowseList="chl_oc2 chl_oc4"
L4ProdList="chl_oc2 chl_oc4"
L4BrowseList="chl_oc2 chl_oc4"
swfProcess $1 $0
```

The above will give us enough to process the data and product results. Next we'll obtain a SeaWiFS data file, uncompress it and use it as input to our script and examine the results. If we are happy, we'll then place the script into to areas directory.

2.7. Testing Our Script

The script we just created will be called by the APS as if we typed the following command:

```
$ SwfGulf S2001005175131.L1A_HNAV
```

Since, the script must be executable, we run the command:

```
$chmod 755 SwfGulf
```

In fact, you can test your script prior to placing it in the APS by (under the Korn/Bourne shell) doing the following commands:

```
$ . ~/aps_v2.8/aps.conf
$ ./SwfGulf S2001005175131.L1A_HNAV
SwfGulf    : checking if file covers gulf ... yes.
SwfGulf    : converting L1 to L2 ... done.
SwfGulf    : converting L2 to L3 ... done.
SwfGulf    : making browse imagery ... done.
SwfGulf    : adding products to data base ... done.
SwfGulf    : running composites ... done.
```

```
SwfGulf     : completed.
$
```

After this completes, we can examine the browse images by change to the directory created by APS for this. If we assume that the `IMAG_BASE` variable was set to `/home/aps/browse` in the `aps.conf` file, then we can examine the Level–3 JPEG files with (or use any other graphics display program):

```
$ cd ~/browse/lvl3/seawifs/2.4/Gulf/2001/jan
$ kview S2001005175131.L3_HNAV_GOM_chl_oc2.jpg
```

Where did that directory come from? APS creates is automatically. By default, APS will put all the HDF product files in the `rs` directory and the JPEG images in the `browse` directory. If the defaults were selected during the installation, this would be subdirectories of your home account. That is, `/home/aps/rs` and `/home/aps/browse`.

The remaining sublevels are created automatically by APS and generally following the pattern `level/sensor/version/area/year/month/`. This pattern is true for all Level–3 data (HDF or browse images). For Level–4, the basic pattern is the same, except we add a name to for the temporal nature of the composite and delete some subdirectories that are not needed. So, for Level-4, the directories are created for example: `lvl4/seawifs/2.4/Gulf/daily/2001/jan`, `lvl4/seawifs/2.4/Gulf/weekly/2001`, or `lvl4/seawifs/2.4/Gulf/yearly`.

You will notice that the SeaWiFS data is stored in a `2.4` directory. This version is a *algorithm processing* version and may or may not reflect that software version number. This was done because the APS can handle many different sensors, whose processing algorithms may or may not change with the software versions. For APS v2.8, the SeaWiFS data is at version 2.4, the MODIS data is at version 1.0, and the AVHRR data is at version 3.0. For the next public release of APS (v2.8), the SeaWiFS will most likely be called version 4.0 to reflect the use of the NASA 4th reprocessing codes. It is also like that MODIS will changed, but whether to 1.2 or 2.0 is undecided.

Thus far, we have been able to use the APS software to run the standard SeaWiFS processing for our area of interest. If we have only a few files, it would make sense to run the above commands for each SeaWiFS file. However, the strength of APS is that we don't just have a few files – we have hundreds or thousands of files. Or, we have a receive station that is continually capturing more data. Then running them by hand does not make much sense. The next section will show how to start processing the data automatically.

Another option for users who do not require automation (because you order and download a bunch of files from the DAAC and then just process them), is the `repro` option for the `aps` script. This option allows the user put a list of files into a UNIX text file and process only those files and areas that they wish.

3. AUTOMATION

This section will take what you learned in the previous section "GETTING STARTED" and automate it using the Automated Processing System. Here you will learn:

- how the automated processing system works
- how to process new areas

3.1. How The APS Works

The APS system uses two concepts to allow for automation.

- directory structure
- script monitoring

3.2. Directory Structure

For directory structure, the APS requires several directories that are similar to any office worker's desk. There is an basket labeled "in" where letters, orders, etc. are placed for the worker to process. There is a work area, usually the desktop, where the worker processes the orders, letters, etc. using different protocols depending on what regions, or areas of the country the order covers. And finally, there is an basket labeled "out" which holds the completed orders.

These four directories are the heart of the system. But other directories are required. A worker may have a file cabinet which he must consult to obtain tables of information or data to process the orders. He may also need various tools, which he keeps in a box, or bin, to process the orders. If an order is incomplete or wrong, he might have a basket labeled "err" to hold these until he can consult with the boss for corrections.

Finally, if a really big order is coming from a colleague down the hall, the colleague might need a temporary place to put portions of the order before placing the complete order in the "in" basket. Several trips down the hall may be required by the colleague to get the complete order. If he places an incomplete order in the "in" basket, the worker might start trying to work on it before its complete. The worker would throw it out!

In terms of the APS, the worker's office is located in the top-level directory (usually `aps_v2.8`). The subdirectories of this directory are: `in`, `work`, `out`, `bin`, `data`, `err`, `ftp`, `areas`, and `src`.

The basic flow of data (that is, the physical *movement* of the actual data files) is shown in Figure 1. Of special note, is the requirement that the `ftp` and `in` directories must reside on the same disk. It is possible to set up the directory structure such that the other directories are on seperate disks. However, normally all directories exist on the same disk.

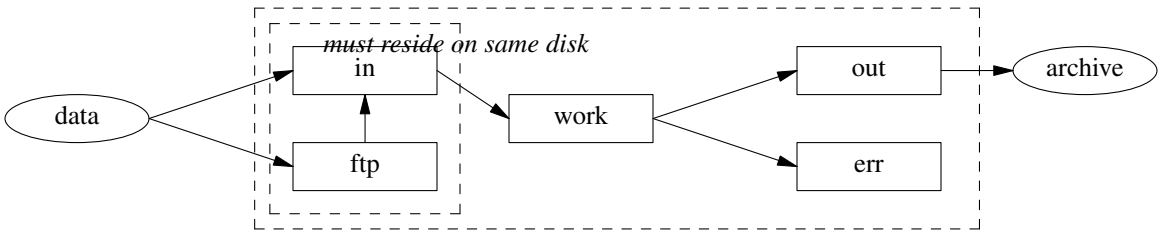


Figure 1. Basic Data File Movement of Level−1 Data

3.3. Script Monitoring

The worker of the APS is the `aps` script. Once started the script will examine the `in` directory for data files, process a single data file for each script in the `areas` directory, and place the data file in the `out` directory. After processing all files in the `in` directory, the script goes to sleep for one minute, after which it performs the above steps again. The script will do this forever. See **aps** (1) for more information.

Figure 2, shows how the execution and Level−1 data file locations correspond. Before, the input Level−1 data file is moved from the `in` directory to the work, a *optional* pre-processing script may be executed. This provides an oppurtunity for the Automated Processing System to do some work on the file before all the areas scripts are run.

Next, each script is executed sequentially on the input data file which now resides in the `work` directory. This allows the various scripts to create temporary files in a clean place. The scripts run various programs located in the `bin` directory, which in turn may read various data files located in the `data` directory.

Finally, the Level−1 data file is moved either to the `err` or `out` directory depending on whether any of the scripts failed abnormally. If the execution was flawless, then the Level−1 data file is moved to the `out` directory where the optional post-processing script will be run. Usually, this post-processing script is designed to remove the Level−1 file from the `out` directory and store it in some type of archival system.

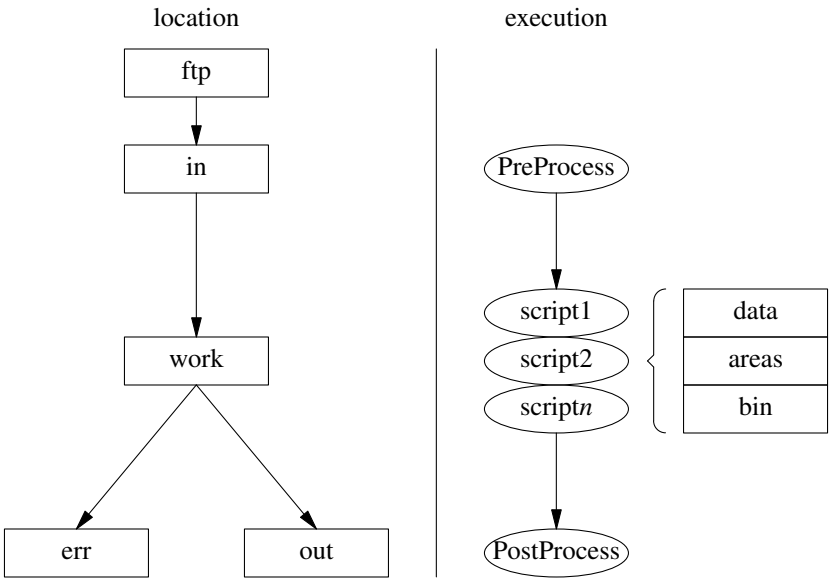


Figure 2. File Location and Execution Flow

3.4. Adding an Area

In the previous section, "GETTING STARTED" we developed the `SwfGulf` script to help us generate an image database of the Gulf of Mexico. That script required a SeaWiFS Level−1A data file as input.

We will add this map to the default file containing all maps, `maps.hdf`, which is located in `$AUTO_DATA` directory.

```
$ hdf gulf_map.hdf copy $AUTO_DATA/maps.hdf gulf
```

To start processing the Gulf of Mexico, simply copy or move the script into the areas directory *and give it execute permissions*. As long as the script is located here, the APS will process incoming data for that area. To stop processing the Gulf of Mexico, you can move the script to another directory, delete it, or make the script non-executable. For example:

```
$ cp SwfGulf $AUTO_DIR/areas
```

will add the `SwfGulf` script to the area directory. To remove it we may do:

```
$ cd $AUTO_DIR/areas
$ chmod 644 SwfGulf
```

– or –

```
$ cd $AUTO_DIR/areas
$ mkdir extra_areas
$ mv SwfGulf extra_areas
```

3.5. Processing Data

At NRL Remote Sensing a SeaWiFS receiving station is used to capture every pass over the Gulf of Mexico. This yields a steady stream of one or two passes (80 MBs of data) a day. After a pass is captured, it is converted to a NASA Level−1A SeaWiFS data file and moved into the `in` directory of another NRL host. The APS is running on this second host. Once the APS *sees* the pass in the `in` directory it is processed.

To process new SeaWiFS data for the Gulf of Mexico, one may simply move the SeaWiFS data files to the `in` directory. However, the aps also provides another method for reprocessing. See **aps** (1) for more information.

4. APS SITE CONFIGURATION

This section will cover some of the other standard procedures deployed in a fully operational system that were not covered previously, like changing your desired list of standard products, and changing the location of the database, etc.

4.1. Changes to the Standard Processing

In this section we will quickly go over some of the more common changes in the ‘‘areas’’ scripts. For a complete list of options see **avhScripts** (1), **mosScripts** (1), and **swfScripts** (1). The standard processing steps are also documented in those man pages.

4.2. Use of apsPreProcess and apsPostProcess

By default, the APS was designed to take Level−1 data and pass it through all the areas scripts and place the Level−1 data in an output directory. Any Level−3 data produced by the respective areas scripts were placed into a database by said script. However, being a receive center, we might also need to convert Level−0 data to Level−1. This conversion would take place so that the Level−1 data was left in the `in` directory and the Level−0 was thrown away. This was accomplished by using a `apsPreProcessing` script.

Additionally, the `apsPreProcessing` script was used to change the permissions on the incoming data so that the APS software was guarenteed to have read/write permissions on the data.

As just noted, the original plan for the Level−1 data was to just place it in the `out` directory. Unfortunately, this directory would eventually fill up the disk! So, the `apsPostProcessing` script was installed to place the Level−1 data on some type of archive system. Our original archive was a tape backup system that has recently been replaced by very large RAID system. The `apsPostProcessing` script would automatically build a standard directory structure based on the data type and time of the Level−1 data and transfer it to the archive disk. Additionally, this script would compress the data.

4.3. Interfacing with a TeraScan System

The TeraScan system has a method of performing post-processing on the incoming data once it is captured. It would be possible to use the individual programs and shell scripts that make up the Automated Processing System and this post-processing mechanism to produce the output files needed. However, at NRL we prefer to have the TeraScan system dedicated to its number one responsibility − data reception. This host will then feed the data (via NIS) to a second system running APS.

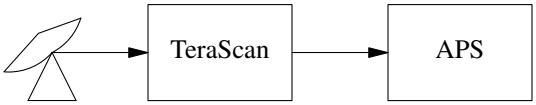


Figure 3. Interfacing with a TeraScan System

To use the system in this manner, a post-processing script is still used, but its primary task is to copy the data from the receive station to the APS system. Included with the APS distribution are example scripts that can be found in the `share` directory. Of important note in these scripts is that (a) the AVHRR Level−0 data is converted to a NESDIS Level−1b using the TeraScan program `flac` and that (b) the SeaWiFS Level−0 data is transferred in TeraScan Level−0 format. In the case of (a), the APS system is set up to expect that type of input. In the case of (b), the APS system has the necessary software to convert the TeraScan Level−0 format to a NASA Level−0 format as required by the NASA Level−1A generating software. A user without a receive station will not be concerned with these matters.

4.4. Sending SeaWiFS Level−1A data to Goddard

At NRL, we are a designated receive site for SeaWiFS data and are, therefore, required to send the data to NASA/Goddard Space Flight Center. The installation procedure covers the required parameters for sending SeaWiFS Level−1A data to Goddard. These are defined in the APS configuartion file, `aps.conf`.

The standard procedure for transfering data to Goddard is to begin by converting the Level−0 data to Level−1A. During this process, an ASCII file is created (with a `.meta` extension) that is to be mailed to a specific address at Goddard. This e-mail message will contain all the information needed by the SeaWiFS Project's software to automatically ftp to your local machine and get the data.

The e-mail handshaking consists of the following steps:

local	→	Goddard	E-mail from Level−1A processor sent.
local	←	Goddard	Goddard e-mails a acknowledgement
local	→	Goddard	Goddard ftps to your site to get file
local	←	Goddard	Goddard e-mails a acknowledgement of transfer
local	←	Goddard	Goddard e-mails a acknowledgement of validity of data

4.5. Obtaining SeaWiFS Level−1A data from Goddard by Subscription

The Goddard DAAC has a subscription capability in which SeaWiFS data will automatically be staged on their systems for pickup. When the data is ready the DAAC will issue an e-mail indicating where the file can be downloaded.

Included in the APS system is software that can handle this option.

4.6. Changing the Logo and text string on the browse image

The browse images contain a logo in the bottom right corner and also three text strings which can be user defined. The strings on the left can not be modifed. The logo displayed is read from the JPEG formatted file `$AUTO_DATA/logo.jpg`. The user should name their logo as such. The logo file should generally be small about 48 by 48 pixels. The APS software will not reduce or manipulate it in any way.

To define the three strings, the user can modify the UNIX ASCII file `$AUTO_DATA/img-Browse.opt`. This file may have the standard shell script comments (that is, lines that start with #). The strings are defined using the keywords (`string1`, `string2`, and `string3`) for the top, middle, and bottom strings. The equal sign (=) that separates the keyword and its value should have not spaces between the two. The string does not have to be in quotes. For example,

```
# My image browse options
string1=UNCLASSIFIED
string2=Approved for Public Release
string3=Distribution Unlimited
```

4.7. Reading APS products in SeaDAS

There are two types of data files produced by programs in the APS which can not be directly read into SeaDAS. The first is a flat−binary format similiar to the PC−SEAPAK format, except that the restriction of data type and image size has been lifted. The second is an HDF formatted file defined by APS.

Several `.pro` files are available in the `share` directory that can read these two formats. To implement these, begin by copying them into the `idl_lib` directory of your SeaDAS installation. Next, you will need to modify the the SeaDAS file `seadispx.pro` (be sure to keep a backup). The file creates the menu for the 'Load' button under SeaDISP. The procedures below *will not* work with the embedded version of SeaDAS.

To add the NRL `.pro` files to `seadispx.pro` use your favorite UNIX text editor and add two lines for each format. The first new line must go in the function `SEADISPX_EVENT` defined at the top of `seadispx.pro` in the case statement for event.value of the 'Load Menu' (approximately at line 50). Insert either or both of these lines:

```
'NRL (HDF)':            SDP_LOAD_NRL, GROUP=event.top
'NRL (SeaPAK)':         SDP_LOAD_NRL_SPK, GROUP=event.top
```

The second location in `seadispx.pro` is in the function `SEADISPX` where `menu1` is created (approximately at line 180). At this location insert either or both of these lines:

```
{ CW_PDMENU_S, 0, 'NRL (HDF)' },     $
{ CW_PDMENU_S, 0, 'NRL (SeaPAK)' }, $
```

When you restart SeaDAS, you will see either or both of the above entires under the 'Load' pull-down menu for the program SeaDISP.

4.8. Setting up a PostgreSQL database

4.8.1. Creating Databases And Setting Up Tables

Log in as postgres after installing the database software.

See your systems' administrator for questions about installing the database software, PostGreSQL.

The first thing we must do is create a database. Generally, you should create user names for individuals who will use the database, and a "root" database user(Using the postgres system user is ok in which case you don't need to create a special user for that). The command "createuser username" will create a user with name->username. It will ask if the user can create databases, and if the user can create new users. Answer these as appropriate for the user you are creating.

Next we should create a database. Use the command "createdb dbname" to create a database with name dbname. The name should be "rs_lvl#", where # is the Level of the datafiles we will keep(lvl3,lvl4,etc). In order to set up the tables and permissions, we must log into the database. First we need to locate the text-file that will create the database we want. They are named like the dbname("setupRS_LVL3", "setupRS_LVL4", etc). Use the command "psql -Uusername dbname" to log into dbname as username. at the prompt, use "\i textfile". This will read the text file which in turn creates a Group g_aps, an aps User, a web7333 User, and all the tables. The 'aps' user is used by the aps system to catalog and modify files by default, but if you installed the APS system under a different user account, you should create an identical username in SQL. Use the command "CREATE username IN GROUP g_aps;". Any user in the Group g_aps will have the same permissions as the aps User. The web7333 user is given only Read permissions on the tables except the webparameters table which it may modify through other programs.

Example setup of the rs_lvl3 database:

```
bash-2.04$ createuser admin
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) y
CREATE USER
bash-2.04$ createdb rs_lvl3
CREATE DATABASE
bash-2.04$ psql -Uadmin rs_lvl3

Welcome to psql, the PostgreSQL interactive terminal.
Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

rs_lvl3=#\i setupRS_LVL3
rs_lvl3=#\q
```

Postgres will spit out a bunch of information after the \i command. This is all normal.

4.8.2. Using PostgreSQL On The Internet

One of the many features of using a database system is the fast and accurate query returns. PostgreSQL may be used in several Programming Languages. Many of these can be run as CGI scripts on the Internet, and even some(PHP) may be embedded right inside an HTML file. This provides a convenient and fast way to share product information through the Internet. Anything from finding product images and displaying them to getting detailed information about a file can be done in this manner. Examples of what can be done through the Internet specifically related to image querying is on the NRL web site at "http://www7333.nrlssc.navy.mil/imagery/imageryMenu.hmtl". All the querying and displaying of the results is done through CGI programs written in C. JavaScript is used on the Latest pages for display purposes, but the images are found and placed in a JavaScript array through a C program.

4.8.3. Final Notes On The Database

Be very careful in your statements. SQL is merciless. If you type: "DELETE FROM main;", you will delete all information. Make sure you use the WHERE clause to get ONLY what you want to change. We suggest using the SELECT command to first view exactly what you will change.

You may obtain PostGreSQL documentation from http://www.postgresql.org.

5. INSTALLING APS

The APS installation comes with a `setup` program to make the installation as easy as possible. To begin, it is recommended APS be run as a normal user. It might be advantageous to create a separate user to handle this. In the example below, the user is called "aps" *Note:* This software will only run on a SGI running IRIX 6.5 or a PC running Linux with glibc 2.x and kernel 2.2.x.

1. Login as the aps user. We'll assume that `/home/aps` is the home directory.
2. Insert the CD-ROM labeled "APS v2.8"
3. `cd /CDROM/aps_v2.8`
4. `./setup`

```
Installation and Setup Script for APS v2.8
for i686-pc-linux-gnu


The binaries require:     120 MBs
The data files require:   620 MBs
The source requires:       85 MBs

The input data requirements are sensor/user dependent
It is recommended that you have more than 1 GB.
Generally, an 30 GB drive is required for APS.

The database and browse directories need to be large
enough for your requirements.

------------------------------------------------------------

The directory given here will be the top-level install-
ation directory.  A directory called 'aps_v2.8' will
automatically be created underneath this directory.

Location to install APS [/home/aps] <return>

------------------------------------------------------------"

The directory give here will be the top-level directory
which will contain your HDF database of mapped products.

The database is a simple directory structure which
is setup as follows (by default):

    $DATA_BASE/level/sensor/version/area/year/month

where,
    level is 'lvl1', 'lvl3', etc.
    sensor is 'seawifs', 'avhrr', etc.
    version is $VERSION
    area is the name of the image map (for example: 'MissBight')
    year is the year
    month is the month

Location to put the data base [/home/aps/rs] /rs<return>
```

```
-------------------------------------------------------

The directory give here will be the top-level directory
which will contain your browse images (the .jpg files).

The database is a simple directory structure which
is setup as follows (by default):

    $DATA_BASE/level/sensor/version/area/year/month

where,
    level is 'lvl1', 'lvl3', etc.
    sensor is 'seawifs', 'avhrr', etc.
    version is $VERSION
    area is the name of the image map (for example: 'MissBight')
    year is the year
    month is the month

Location to put the browse imagery [/home/aps/browse]
```
web.satellite.org:/home/httpd/browse<return>

```
-------------------------------------------------------

The directory given here will be the top-level directory
which will contain your ancillary data files.

Location to find ancillary data (? - info) [/home/aps/ancil]
```
/export/rs/ancil<return>

```
-------------------------------------------------------

Enter a string for your Agency or Organization:
```
Satellite Inc.

```
-------------------------------------------------------

Do you plan to process SeaWiFS Level-0 data (Y/N) [N]
```
Y<return>

```
-------------------------------------------------------

You must have a connection to the internet to automatically
obtain the navigation file required for SeaWiFS Level-0 to
Level-1A processing (via FTP).  If not and you plan to process
SeaWiFS Level-0 to Level-1A you must obtain the 'elements.dat'
file on your own.

Automatically get 'elements.dat'? (Y/N) [N]
```
Y<return>

```
-------------------------------------------------------

Do you wish to receive e-mail whenever a Level-0 file
is received that was not properly decypted, enter a
proper e-mail address

User to get e-mail for encrypted L0 files? [aps@torch]
```
<return>

```
-------------------------------------------------------

Do you plan to send SeaWiFS data to Goddard in Near
Real Time via FTP? [N] Y<return>

You must contact Goddard and set up some variables in the
APS configuration file (/home/aps/aps_v2.8/bin/aps.conf)

You should set up a goddard account and goddard group.  We
additionally place a L1A directory in the goddard home
account.  You must place the goddard account on a host
running ftpd.  You must then send Goddard the host/username/
password so that they can access the data.

-------------------------------------------------------

Enter your three letter code [NAV] <return>

-------------------------------------------------------

Enter the ftp directory where you will place the L1A files
(Note: this directory must be NFS mountable from the machine
which will run the APS): [/home/ftp] /export/goddard/L1A<return>

-------------------------------------------------------

Enter the ftp host where Goddard will obtain the file
[modis.satellite.org] ftp.satellite.org<return>

-------------------------------------------------------

Enter the directory under the ftp account where Goddard will
obtain files: [pub/aps_v2.8] ./L1A<return>

-------------------------------------------------------

Enter the ftp notify address [aps@modis.satellite.org]
aps@mail.satellite.org<return>

-------------------------------------------------------

This version of APS has the capability to create an SQL
database or catalog of all data processed.  With this
database the user can quickly search for images that cover
user-specified time and region components.

To use this option, the user must create and run a PostgreSQL
server.  Instructions of setting this up and included in the
user's manual.

Do you want to setup an SQL database? (Y/N) [N] Y<return>

-------------------------------------------------------
```

```
Enter the host where the SQL server will run on
[localhost] <return>

------------------------------------------------------------

Do you want to unpack the source? (Y/N) [N] <return>
Making Directory /home/aps/browse

Unpacking base...
Unpacking i686-pc-linux-gnu binaries ...

Fixing the aps.conf file...

Installation and Setup is complete!

INSTALL_DIR = /home/aps
AUTO_DIR    = /home/aps/aps_v2.8
DATA_BASE   = /home/aps/rs
IMAG_BASE   = web.satellite.org:/home/httpd/browse
ANCIL_DIR   = /export/rs/ancil

------------------------------------------------------------
It is highly recommended that you examine the configuration
file to verify that all parameters are correctly defined.  Your
config file is: /home/aps/aps_v2.8/bin/aps.torch.conf
------------------------------------------------------------
```

5. To start APS type:

```
$ rm /home/aps/aps_v2.8/bin/.aps.pid
$ /home/aps/aps_v2.8/bin/aps init
```

Note: As data is processed a *LOT* of debugging information will start scrolling on the screen. This is normal. You may decide to leave the window up so you can watch things, or you can exit the shell window. The debugging output will then just go to the bit bucket.

6. LIST OF APS PROGRAMS

The following list displays documented and *undocumented* programs found in the APS v2.8 `bin` directory. Those programs that are documented are done so in the form of UNIX man pages. Additionally a majority of these programs have a `--help` option which will print a small usage message.

6.1. Accessing the man pages

To access the `man` pages, the environment variable `$MANPATH` may need to be appended to include their location. The default location is the `man` directory where APS has been installed. For example, if APS was installed in the `/home/aps/aps_v2.8/` directory, then these two lines should be added to your `.profile` for Bourne, Bash, and Korn Shell users:

```
MANPATH="$MANPATH:/home/aps/aps_v2.8/man"
export MANPATH
```

For C shell users, the following line should be added to your `.cshrc` file:

```
setenv MANPATH "$MANPATH:/home/aps/aps_v2.8/man"
```

6.2. Core Commands

aps Main driver program for APS.

apsCronProcess
: Executable script to handle any processing required on each polling cycle of APS. This is useful for any tasks the user wants to execute like routinely checking databases, etc. At NRL, it is used to retreave SeaWiFS Level−0 data file from three regional centers.

apsPostProcess
: Executable script to handle post processing, if any, of the input data file after all the ‘‘areas’’ scripts have been run on the file. This is useful for any tasks the user might wish to perform on the file, like enter it into a catalog, copy it to an archive system, or make a browse image of it. At NRL, it is used to move the data to out archive system automatically.

apsPreProcess
: Executable script to handle preprocessing, if any, of the input data file prior to running all the ‘‘areas’’ scripts on the file. This is useful for any tasks the user wants to execute on the input file prior to running an area. At NRL, it is used to convert an input SeaWiFS Level−0 data file to Level−1A data file.

apsScripts.sh
: A collection of Bourne shell script functions for general use (that is, not sensor specific) by a user written ‘‘areas’’ script.

daylight
: Given a time and earth location, determine whether the sun is above or below the horizon.

filefmt
: Determines the format of an given file. It understands many remote sensing data formats.

gregor
: Return the month and day given a year and day of year or vice versa.

hdf Performs general manipulations (copy, list, dump) of data arrays (Scientific Data Sets or SDSs) stored in an Hierarchical Data Formatted (HDF) file.

maps
Performs general manipulations (create,delete,save) of image maps.

6.3. AVHRR Specific Commands

avhArea
Determines the file extents of a NESDIS Level−1B data file's coverage of an image map.

avhClouds
Produces a bit image of up to five tests used to determine cloud contaminated pixels.

avhDump
Dumps AVHRR video data from a NESDIS Level−1B file.

avhImage
Creates a graphics image from a NESDIS Level−1B file. It can handle several output formats. Use the "--help" option or see **avhImage** (1) to see which formats it supports.

avhInfo
Queries information about a NESDIS Level−1B file.

avhIngest
Produces percent albedo and brightness temperatures from AVHRR data in a NESDIS Level−1B file.

avhScan
Dumps scan line information from a NESDIS Level−1B file.

avhScripts.sh
A collection of Bourne Shell script functions specific for AVHRR data which can be used in a user written areas script.

avhSST
Produces sea surface temperature (SST) images from brightness temperature images.

avhSwapL0
Swaps bytes in a TeraScan L0 formatted file.

avhTurbid
Produces beam attenuation (c_660), diffuse attenuation (K_PAR) and total sediment (suspend) products from percent albedo images.

6.4. MODIS Specific Commands

modArea
Determines the file extents of a NASA MODIS Level−1B data file's coverage of an image map.

modcol_nir

Performs the atmospheric and radiometric corrections on MODIS Level−1B data to produce many bio−optical products including estimates of remote sensing reflectance for the first seven MODIS ocean bands (8-14), diffuse attenuation at 532 nm using a 488/551 ratio, chlorophyll *a* concentration using the SeaBAM OC3M, Clark, and Carder algorithms, total absorption at first six bands using Arnone, Carder, and QAA algorithms, backscattering at 443 and 555 using Arnone, Carder, QAA algorithms, detrital/CDOM absoprtion at 412 using Carder and QAA algorithms, phtyoplankton absorption at 443 using Carder and QAA algorithms, beam attenuation at 670 using Carder and QAA algorithm, horizontal and vertical diver visibility, cloud albedo at 865 nm, true color image, and a product consisting of 32 flags indicating various conditions like atmospheric correction errors, land, Lt > knee, negative water leaving radiance, low normalized water leaving radiance at 555, etc. See **modcol_nirfR (1) for the complete list. The program includes NIR correction methods from Arnone.**

modGetL1B

Extract channel data from a MODIS Level-1 file and put into an APS type HDF file.

modInfo

Queries information about a MODIS Level-1 file.

modRGB

Create a true color product from a MODIS Level-1 file.

modScripts.sh

A collection of Bourne Shell script functions specific for MODIS data which can be used in a user written areas script.

modsst

Performs the atmospheric and radiometric corrections on MODIS Level−1B data to produce sea surface temperature products.

6.5. MOS and SeaWiFS Specific Commands

mosl1corr

Corrects a MOS Level−1B file for straylight.

mosInfo

Queries information about a MOS Level−1B file.

mosScripts.sh

A collection of Bourne Shell script functions specific for MOS data which can be used in a user written areas script.

MSl12

Performs the atmospheric and radiometric corrections on SeaWiFS Level−1A or MOS Level−1B data to produce many bio−optical products including estimates of remote sensing reflectance for the first six SeaWiFS/MOS bands, diffuse attenuation at 532 nm using a 490/555 ratio, chlorophyll *a* concentration using the SeaBAM OC4, Stumpf, Carder, and QAA algorithms, total absorption at first six bands using Arnone, Carder, and QAAalgorithms, backscattering at 443 and 555 using Arnone, Carder, and QAA algorithms, detrital/CDOM absoprtion at 412 using Carder and QAA algorithms, phtyoplankton absorption at 443 using Carder and QAA algorithms, beam attenuation at 670 using

Carder and QAA algorithms, horizontal diver visibility, albedo at 768 nm, true color image, and a product consisting of 32 flags indicating various conditions like atmospheric correction errors, land, Lt > knee, negative water leaving radiance, low normalized water leaving radiance at 555, etc. See **MSl12** (1) for the complete list. The program includes NIR correction methods from Arnone, MUMM, Siegel, Stumpf, as well as a correction scheme of Stumpf's known as the ''412 iteration''

swfArea
Determines the file extents of a NASA SeaWiFS Level−1A data file's coverage of an image map.

swfArnone
Generates total absorption and backscattering products at the first six SeaWiFS wavelengths from SeaWiFS remote sensing reflectance data.

swfCarder
Generates total absorption, absorption due to detris and gelbstoff, absorption due to phaepigments, backscattering, and chlorophyll *a* concentration products from SeaWiFS remote sensing reflectance data. The absorption and scattering products are generated for the first six SeaWiFS wavelengths.

swfGetElements
Obtain the orbital elements file required by swfL1agen over the internet.

swfInfo
Queries information about a SeaWiFS Level−1A file.

swfL1agen
Creates a NASA SeaWiFS Level−1A file from a NASA SeaWiFS Level−0 file.

swfScripts.sh
A collection of Bourne Shell script functions specific for SeaWiFS data which can be used in a user written areas script.

swfSeadas
Creates the default SeaDAS products of chlorophyll *a* and K_490 using the nLw from an input file.

swfStumpf
An iterative correction method based on work from Rick Stumpf.

swfTeraL0
Converts a TeraScan SeaWiFS Level−0 formatted file to a NASA SeaWiFS Level−0 formatted file.

swfVisibility
Creates an estimate of diver visibility from beam attenuation using an algorithm from McBride.

6.6. Image Specific Commands

imgBathy
Create a ''bathymetry'' SDS in an HDF file.

imgBrowse
Creates quick-look browse images for a Level−3 or Level−4 product. It automatically draws

coastline, grids, color scales, and annotations. It can output files in JPEG, PNG, or TIFF format.

imgConvolve
: Convolve an SDS image.

imgDiff
: Generate a difference image for an SDS in two separate files.

imgDump
: Dump an SDS from an HDF file to a selected ASCII format.

imgFlags
: Creates an NSIPS image file from an "flags" SDS in an HDF file.

imgLandMask
: Create a ‘‘land’’ SDS in an HDF file.

imgMakeLatLon
: Create the ‘‘latitudes’’ and ‘‘longitudes’’ SDSs in an HDF file.

imgMap
: Image warping to one of 25+ USGS map projections from a 2−D HDF SDS given a control points array covering the input image.

imgMean
: Generate a composite from a list of Level-3 or Level-4 files for an arbitary number of products. Limited to 31 input files due to an HDF library limitation.

imgMean.sh
: A script that implements imgMean for handling more than 31 input files.

imgMedian
: Perform a median filter an SDS image.

imgRead
: Read point data from a series of SDSs in an HDF file given a series of individual points.

imgReformat
: Tiles and compresses an HDF 2−D SDS.

imgSDStoImg
: Convert a series of SDSs from an HDF file into several other formats, like NSIPS, PC-SEAPAK, ENVI, GeoTiff, etc.

imgTSeries
: Perform a statistical average of a region of interest.

spkToSDS
: Converts a PC-SEAPAK formatted file to an SDS in an HDF file.

6.7. Miscellaneous Commands

mice A preprocessor for compiling U. of Miami mice code.

ratfor
: A Fortran 77 preprocessor for compiling U. of Miami RATFOR code into Fortran 77 code.

ratf90
: A Fortran 90 preprocessor for compiling U. of Miami RATFOR code into Fortran 90 code.

smfcompile
: Code use by the SDP Toolkit for converting messages into C and Fortran headers.

6.8. File Formats

aps.conf
: Configuration file for aps script.

7. FREQUENTLY ASKED QUESTIONS

For the following questions, it is assumed that the APS software has been properly installed. See "INSTALLING APS" in for more information. In particular, these examples assume that the APS software was installed in `/home/aps`.

Q. Is the aps running?

A. In the processing table you will see an entry for it.

```
$ ps -ef | grep aps
 aps       1325       1225  0 11:03:46 pts/3   0:00 ps -ef
 aps       1339       1259  0 11:03:05 pts/0   0:00 sleep 60
 aps       1340       1225  0 11:03:46 pts/3   0:00 grep aps
 aps        938        935  0 10:01:13 pts/2   0:00 -ksh
 aps        961        867  0 10:11:30 pts/0   0:00 -ksh
 aps       1225        993  0 10:58:31 pts/3   0:00 -ksh
 aps       1259          1  0 11:02:03 pts/0   0:00 /bin/ksh
/home/aps/aps_v2.8/bin/aps begin
```

The last entry shows the aps running. It's processing ID is 1259. It is currently asleep (see second line of `ps` output).

Q. The aps won't start. What do I need to check to get it running?

A. First, goto to the previous question and make sure it is not ALREADY running! If you are sure, then go to `/home/aps/aps_v2.8/bin` directory and look in the file `.aps.pid`. Try to `grep` for the number in there:

```
$ ps -ef | grep `cat .aps.pid`
 aps       1351       1259  0 11:07:05 pts/0   0:00 sleep 60
 aps       1259          1  0 11:02:03 pts/0   0:00 /bin/ksh
/home/aps/aps_v2.8/bin/aps begin
```

In this case it *is* running. But suppose it is not, then your machine died before the script could remove this file. You may either reboot, or do the following two commands.

```
$ rm /home/aps/aps_v2.8/bin/.aps.pid
$ /home/aps/aps_v2.8/bin/aps init
```

Q. How do I determine what areas are being processed?

A. List the scripts located in the `areas` directory. Only those files that are *regular executable* files will be run. For example,

```
$ ls -F ~/aps_v2.8/areas
AvhEastSea*         AvhPusan*           SwfMissBight
AvhGulfofMexico*    AvhSubPolarFront*   SwfPusan*
AvhMissBight        SwfEastSea*         SwfSubPolarFront*
AvhNYBight*         SwfGulfOfMexico*    extra_areas/
```

Here, the EastSea, Gulf of Mexico, Pusan, and SubPolarFront regions are being processed for both AVHRR and SeaWiFS data. The NYBight is being processed for AVHRR only. The MissBight region is not being processed and `extra_areas` is a directory. See the **–F** option in **ls** (1).

Q. How do I turn off a region?

A. Move the script that processes the desired region out of the `areas` directory or make it non-executable.

```
$ cd ~/aps_v2.8/areas
$ ls -F
AvhEastSea*         AvhPusan*           SwfMissBight
AvhGulfofMexico*    AvhSubPolarFront*   SwfPusan*
AvhMissBight        SwfEastSea*         SwfSubPolarFront*
AvhNYBight*         SwfGulfOfMexico*    extra_areas/
$ mv *Pusan extra_areas
$ ls -F
AvhEastSea*         AvhSubPolarFront*   SwfSubPolarFront*
AvhGulfofMexico*    SwfEastSea*         extra_areas/
AvhMissBight        SwfGulfOfMexico*
AvhNYBight*         SwfMissBight
```

or

```
$ chmod 644 *Pusan
$ ls -F
AvhEastSea*         AvhPusan            SwfMissBight
AvhGulfofMexico*    AvhSubPolarFront*   SwfPusan
AvhMissBight        SwfEastSea*         SwfSubPolarFront*
AvhNYBight*         SwfGulfOfMexico*    extra_areas/
```

Q. How do I turn on another region?

A. Move the script that processes the desired region into the `areas` directory or make it executable.

```
$ cd ~/aps_v2.8/areas
$ ls -F
AvhEastSea*         AvhSubPolarFront*   SwfSubPolarFront*
AvhGulfofMexico*    SwfEastSea*         extra_areas/
AvhMissBight        SwfGulfOfMexico*
AvhNYBight*         SwfMissBight
$ mv extra_areas/*Pusan .
```

or

```
$ chmod 755 *Pusan
$ ls -F
AvhEastSea*         AvhPusan*           SwfMissBight
AvhGulfofMexico*    AvhSubPolarFront*   SwfPusan*
AvhMissBight        SwfEastSea*         SwfSubPolarFront*
AvhNYBight*         SwfGulfOfMexico*    extra_areas/
```

Q. Can I stop a region from processing while it is processing?

A. Not easily. To do so you will have to use the `ps` command and determine which programs are running and kill them individually. Unfortunately, it is not an easy task in UNIX to kill a particular task and *all its children*.

Q. How do I determine if the APS is hungup?

A. Again, not easily. You might `cd` to the `work` directory and check any log files there. If you see one you might do a `tail -f` on that file and wait for output. For example, when the `MSL12` program is running it writes messages after every 50 lines processed.

Another method would be to use the **ps** (1) command to check the user time of the program. The AVHRR programs should generally take 10 or less minutes. If the user time is greater than this it is an indication of a hang up. The SeaWiFS programs may take longer to complete, especially **MSl12** (1). This program is very computer intensive and may take twenty to forty minutes to complete depending on your hardware.

A third method may be to look in the `work` directory and use the `ls -l` command to periodically check the file sizes. This will work for the AVHRR processing by examining the `levels*` and `turbid*` files. The HDF file produced by the SeaWiFS and MOS processing may be prebuild to desired output size and may not change in size. However, the time stamp should change as the file is modified.

8. ACRONYMS

AVHRR	Advanced Very High Resolution Radiometer
APS	Automated Processing System
HDF	Hierarchical Data Format
GCP	Ground Control Point
GCTP	General Cartographic Transformation Package
JPEG	Joint Picture Expert Group
MODIS	Moderate Resolution Imaging Spectroradiometer
MOS	Scientific Data Sets
NSIPS	Naval Satellite Image Processing System
SeaWiFS	Sea-viewing Wide Field-of-view Sensor
SQL	Structured Query Language
SDS	Scientific Data Sets

9. ACKNOWLEDGEMENTS

Some of the MODIS processing routines are from the University of Miami and contain the following copyright:

The HDF-EOS library contains the following copyright:

Some of the SeaWiFS processing routines are from SeaDAS which was obtained from NASA Goddard Space Flight Center. No IDL routines from SeaDAS are used in the Automated Processing System. SeaDAS contains the following COPYING file:

2. That modifications which may be considered generally useful, be forwarded so that they might be incorporated into future releases of SeaDAS.

Regarding the commercial use of SeaDAS, be aware that SeaDAS does integrate software from several sources which may include embedded copyright notices and which may have some limitations with respect to commercial sales. Specifically we include the NCSA HDF libraries, the University of Miami's DSP libraries (used only for reading CZCS PST format files), a few instances of source code developed by the University of Miami and used in SeaWiFS data processing, and modified versions of RSI's IDL programs. Anyone considering modifying and selling SeaDAS should read all the embedded copyright and copying notices and proceed at their own risk.

By copying this software, you, the user, agree to abide by the conditions and understandings with respect to any software which is marked with a public domain notice.

ACKNOWLEDGEMENTS:

The main portion of the SeaDAS software was developed by:

1. The SeaDAS Development Group at NASA GSFC, Greenbelt, Maryland
2. The SeaWiFS Project at NASA GSFC, Greenbelt, Maryland.
3. The SIMBIOS Project at NASA GSFC, Greenbelt, Maryland.

The SeaDAS software product also includes:

1. The HDF4.1r1 libraries which were developed by the National Center for Supercomputing Applications (NCSA) at the University of Illinois (UI)

2. Modified IDL procedures originally developed by Research Systems, Inc, (RSI).

/*==*/